Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for providing an adapter or stub as needed for a virtual machine during runtime when said virtual machine executes computer code, said method comprising:

identifying a machine state input parameter for a machine state;

identifying input parameters for a call to compiled code;

mapping the machine state input parameter and the machine state to the input parameters for the call to compiled code;

mapping the machine state and return value to an exit point of an interpreter to compiled code adapter, thereby generating an adapter/stub representation that can be configured as an adapter or a stub for the virtual machine during runtime;

determining during runtime whether to configure the adapter/stub representation as an adapter or as a stub for the virtual machine;

~~configuring~~ generating said adapter/stub representation during runtime as a stub representation that is provided to a compiler for compilation and generating object code based on the compilation when the determining determines to configure the adapter/stub representation as a stub;

determining during runtime whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter when the determining determines to configure the adapter/stub representation as an adapter;

~~configuring~~ generating said adapter/stub representation during runtime as an interpreter to compiled code (I/C) adapter that facilitates translation of a first execution stack used by an interpreter associated with the virtual machine when the determining determines to provide the interpreter to compiled code (I/C) adapter, so that the first execution stack can subsequently be used to execute compiled-code compiled by a compiler associated with the virtual machine; and

~~configuring~~ generating said adapter/stub representation during runtime as a compiled code to interpreter (C/I) adapter that facilitates translation of a second execution stack used for execution of compiled code compiled by a compiler associated with the virtual machine when the determining determines to provide the compiled code to interpreter (C/I) adapter, so that the second execution stack can subsequently be used by an interpreter associated with the virtual machine.

2-8. (Canceled)

9. (Previously Presented) The method of claim 1, wherein the method is performed in response to a determination that the adapter/stub is not stored in an adapter/stub library associated with the computer system.

10. (Previously Presented) The method of claim 9, wherein the determination is performed when compiled code is to be executed by the computer system, and the computer system determines that an interpreter to compiled code (I/C) adapter/stub is required.

11. (Canceled)

12. (Previously Presented) The method of claim 1, wherein the adapter/stub is further operable to update the states of different components of the computer system.

13-16. (Canceled)

17. (Previously Presented) A method as recited in claim 1, wherein said determining of whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter comprises: determining whether one or more bytecodes have been processed by a interpreter.

18. (Currently Amended) A computer readable medium embodied in a tangible form including computer program code for providing an adapter or stub as needed for a

virtual machine during runtime when said virtual machine executes computer code, comprising:

computer program code for generating an adapter/stub representation that can be configured as an adapter or a stub for the virtual machine during runtime;

computer program code for determining during runtime whether to configure the adapter/stub representation as an adapter or as a stub for the virtual machine;

computer program code for ~~configuring~~ generating said adapter/stub representation during runtime as a stub representation that is provided to a compiler for compilation and generating object code based on the compilation when the determining determines to configure the adapter/stub representation as a stub;

computer program code for determining during runtime whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter when the determining determines to configure the adapter/stub representation as an adapter;

computer program code for ~~configuring~~ generating said adapter/stub representation during runtime as an interpreter to compiled code (I/C) adapter that facilitates translation of a first execution stack used by an interpreter associated with the virtual machine when the determining determines to provide the interpreter to compiled code (I/C) adapter, so that the first execution stack can subsequently be used to execute compiled-code compiled by a compiler associated with the virtual machine; and

computer program code for ~~configuring~~ generating said adapter/stub representation during runtime as a compiled code to interpreter (C/I) adapter that facilitates translation of a second execution stack used for execution of compiled code compiled by a compiler associated with the virtual machine when the determining determines to provide the compiled code to interpreter (C/I) adapter, so that the second execution stack can subsequently be used by an interpreter associated with the virtual machine.

19. (Previously Presented) A computer readable medium including computer program code for providing an adapter/stub for a virtual machine during runtime, wherein the adapter/stub can behave as an adapter or a stub for the virtual machine, comprising:

computer program code for identifying a machine state input parameter for a machine state;

computer program code for identifying input parameters for a call to compiled code;

computer program code for mapping the machine state input parameter and the machine state to the input parameters for the call to compiled code; and

computer program code for mapping the machine state and return value to an exit point of an interpreter to compiled code adapter.

20. (Previously Presented) A computer readable medium as recited in claim 18, wherein said determining of whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter comprises: determining whether one or more bytecodes have been processed by a interpreter.

21. (Currently Amended) A computing system, comprising:

at least one processor that operates to:

determine during runtime whether to configure the adapter/stub representation as an adapter or as a stub for the virtual machine;

~~configure~~ generate said adapter/stub representation during runtime as a stub representation that is provided to a compiler for compilation and generating object code based on the compilation when the determining determines to configure the adapter/stub representation as a stub;

determine during runtime whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter when the determining determines to configure the adapter/stub representation as an adapter;

~~configure~~ generate said adapter/stub representation during runtime as an interpreter to compiled code (I/C) adapter that facilitates translation of a first execution stack used by an interpreter associated with the virtual machine when the determining determines to provide the interpreter to compiled code (I/C)

adapter, so that the first execution stack can subsequently be used to execute compiled-code compiled by a compiler associated with the virtual machine; and

~~configure~~ generate said adapter/stub representation during runtime as a compiled code to interpreter (C/I) adapter that facilitates translation of a second execution stack used for execution of compiled code compiled by a compiler associated with the virtual machine when the determining determines to provide the compiled code to interpreter (C/I) adapter, so that the second execution stack can subsequently be used by an interpreter associated with the virtual machine.

22. (Previously Presented) A computing system as recited in claim 21, wherein the at least one processor operates to:

identify a machine state input parameter for a machine state;

identify input parameters for a call to compiled code;

map the machine state input parameter and the machine state to the input parameters for the call to compiled code; and

map the machine state and return value to an exit point of an interpreter to compiled code adapter.

23. (Previously Presented) A computing system as recited in claim 21,

wherein said determining of whether to provide an interpreter to compiled code (I/C) adapter or a compiled code to interpreter (C/I) adapter comprises: determining whether one or more bytecodes have been processed by a interpreter.